An Interop Publication

CONNEXIONS®

The Interoperability Report

June 1995 Volume 9, No. 6

ConneXions — The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published monthly by Interop Company, a division of SOFTBANK Exposition and Conference Company, 303 Vintage Park Drive, Foster City, California, 94404–1138, USA.
Phone: +1 (415) 578-6900
Fax: +1 (415) 525-0194
E-mail: connexions@interop.com

Subscription hotline: 1-800-575-5717
or +1-502-493-3217

ISSN 0894–5926

From the Editor

Many organizations, particularly large corporations, are somewhat reluctant to join the Internet. Their misgivings usually have to do with security. While most companies see the value of Internet access, they worry about other users on the Internet having access to their internal networks. The most common solution to this problem is to install a *Firewall*. According to one expert I consulted, a firewall can be defined as "an administrative chokepoint introduced into a network topology for the purpose of enforcing security policies." We plan to bring you an introduction to firewalls in a forthcoming issue. This month, however, Ted Doty of Network Systems Corporation presents a dissenting opinion with regard to firewall technology in an article entitled "Firewall Heresies." Debate about technical merits is something we always encourage, so send us your reaction, and be sure you copy the author.

The most popular of all Internet applications these days is the World-Wide Web (WWW). We've covered several aspects of Web usage and design in recent issues. This month we look at some of the challenges facing this application. Jon Crowcroft and Mark Handley from University College London describe some of the problems and some possible solutions. Meanwhile, the uses of Web technology are becoming more and more sophisticated. Just today I discovered that I can access my bank account (securely of course) via Wells Fargo's on-line banking application. Very cool!

In our Book Reviews section this month, Dave O'Leary of Cisco Systems looks at two recent books on one of the most complex of all networking topics, namely *routing*. Unlike books about the Internet, there aren't many publications to choose from if you want to learn about routing. It is therefore very encouraging to see two books appear nearly at the same time, both written by recognized experts in the Internet engineering community.

Speaking of recognized experts, I urge you to contact us with suggestions for future articles, and in particular pointers to authors who can write them. Your feedback is always appreciated and as usual you can send it via e-mail to `connexions@interop.com`.

Finally, another reminder that all back issues of *ConneXions* are still available for purchase, that's all 99 of them! You can browse our index of back issues on `http://www.interop.com` or drop us an e-mail message with your postal address and we will send you the index in hardcopy form.

The Firewall Heresies

by Ted Doty, Network Systems Corporation

Abstract

Current firewall implementations, particularly those based on proxy agents, result in slower, more expensive firewalls than other technologies such as packet filtering. The implication of lower performance is that it is not possible to firewall high speed applications like NFS or IPX/NCP using proxies, and by extension create internal firewalls to protect sensitive data such as personnel records that is contained on LANs attached to corporate backbones. In addition, source code examination of proxy-based firewalls is insufficient for demonstrating secure operation; it is argued that proper collection and analysis of audit information is the only method to ensure correctly operating security. Also, since non-IP protocols provide equally effective means of exchanging data, protocols other than the Internet suite of protocols require firewall capabilities. Proxy implementations do not currently provide this. It is argued that firewalls built with sufficiently powerful packet filtering capabilities can provide a superior security solution to proxy-based technology.

Introduction

Firewalls have rapidly emerged as an extremely valuable tool for protecting organizations from the barbarians who roam the Internet. Firewalls have two main virtues: they provide rigorous protection for otherwise difficult to protect computers by acting as a door guard, and they do so in a (more or less) understandable manner. This second point is perhaps the most important, because without assurance of correct implementation, there can be no security.

Firewalls have been based on two fundamental premises: first, access policies must be kept small, because small is verifiable; second, the tools that implement the policy in the firewall must be kept small, because small is understandable. These ideas have been well documented in [4], and can be generally accepted as an excellent starting point for establishing security controls in a network.

However, most firewall architectures have used a technology that drastically limits the usefulness of its security features. The ideas presented in these pages are an attempt to open the firewall discussion to three topics that have previously been ignored. Specifically, the areas of performance, multiple protocols, and cost have been notably lacking from the debate.

Several ideas described here attempt to address these issues. The ideas run against the grain to the Current Wisdom, and are considered profoundly heretical by some. However, these ideas are by no means new; they have been implemented in different areas of technology; software engineering in particular has much to offer. These ideas can lead to a newer, better firewall technology: *Firewall Routing*.

The heresies

In brief, the Firewall Heresies discussed in this article are:

- Performance counts, especially since "internal" firewalling between different groups in an organization can be nearly as important as "external" firewalling between the entire organization and the Internet.
- Examination of source code cannot verify correctness for almost all firewall implementations.
- Proper attention to audit trails can determine errors in the specification or implementation of the access policy, as well as defects in the software enforcing the policy; it is properly considered vital to verification of correctness.

- Current firewall implementations will not support protocol suites other than the Internet protocols.
- Advanced packet filtering in network routers can accommodate both high performance applications and non-IP protocols, in a verifiable manner.

The remainder of this article is devoted to the discussion of each of these heresies. Since each of these (with the possible exception of the first) is diametrically opposed to the Current Wisdom, a justification will be presented for each.

• *Heresy #1: Performance Counts:* Firewalling has been described as constructing a "hard, crunchy shell around a soft, chewy center." This view of the universe suggests that everyone in the soft, chewy center should have equal access to all the data in the "nougat." This is not correct for many cases.

There are at least two types of data that require access controls *within* an organization. The most obvious of these is confidential personnel data. While many other people work for Network Systems, they have no business knowing how much money I make, or how many sick days I took last year, or what types of personal disabilities I might have. In the USA, at least, this information is required to be protected (by law). In other countries, there is at least a moral obligation to protect this data. While gross negligence in the protection of this information may not cause legal or direct financial harm to the organization, there would be a definite cost in terms of employee good will should this information be disclosed due to negligence or malice.

Financial information also needs strong protection. Payroll, accounting, and tax records could be broadly considered "company information." According to the Current Wisdom, this data needs protection only from the barbarians outside the organization. From a risk assessment point of view, this is demonstrably incorrect. The easiest way to put a company out of business is from unauthorized alteration of tax records or accounts receivable. While the risk may be lower for data access within an organization, the potential cost of any security breach is so large that this data requires firewalling.

Given the need for "internal" firewalls, we can no longer say that performance is not important. Unfortunately, most discussions of firewall performance to date have ended with the words "... and anyway, the firewall is faster than your T1 link to the Internet." While this may be sufficient performance to protect the organization's connection to the Internet, it is insufficient for departmental LANs attached to the corporate backbone.

It is no secret that current firewalls are slow, by any measure we apply to networks today. More surprisingly, they are slow for the same reasons that caused poor performance in the Bad Old Days of networking, and which have been well documented [5]. (The author remembers being present at conferences where Gray Beards claimed [with perfectly straight faces] that they could prove that TCP would never run faster than a megabit). The problem of firewall performance is directly attributable to the architecture built using proxies running in application space which require continual context switches and data copying. The reason that routers run at half a million packets per second is that data is never copied. The reason that firewalls run at 2 orders of magnitude less is that data is copied all the time.

continued on next page

The Firewall Heresies *(continued)*

- *Heresy #2: Source Code evaluation does not mean verifiable security:*

```
*/ You are not expected to understand this. */
```

No, this comment is not referring to this section (hopefully). Rather, it refers to the most (in)famous line in the UNIX kernel source code.

The assertion often made in firewalling circles (the Current Wisdom) is that every line of source code must be inspected, in order to ensure bug-free implementation. While source code evaluation is generally accepted as good software engineering practice, sole reliance on code inspection brings up a number of problems, both practical and theoretical.

From a theoretical perspective, it is unlikely that even small software projects can be demonstrated to be bug free. While it is a matter of debate whether firewalls have evolved to this size limit, we find firewalls engaged in a race to add new functionality (such as encryption). If nothing else, the number of states in an *n*-proxy firewall gives rise to serious doubt as to the completeness of any code walk-through. There is doubt [2] that even relatively thorough unit and integration testing can reveal an entire class of bugs: those which are load-dependent.

From a practical point of view, code inspection requires a thorough knowledge of software engineering in general and operating systems in particular. In practice, this means **C** and UNIX. Customers lacking this expertise cannot avail themselves of this option. However, for those that can, how can we trust that the evaluation was performed correctly? Each evaluation is likely to find only a portion of the software defects present in the code. Essentially, we are forced to perform multiple, independent evaluations, each at great expense, and only asymptotically approaching zero defects.

Another practical issue is that as long as this industry remains essentially artisanal in nature, the cost of the finished goods will be high. Indeed, this is what we see today. Since very few organizations can afford a full time firewall guru, firewalls will become wide spread slowly, if at all. At the extreme, we could imagine a race of high-tech *condotierri* who will roam the land installing and re-installing firewalls, without much in the way of oversight or assurance.

• *Heresy #3: Audit trails are the only hope for security:* Given the difficulty in demonstrating correctness via code reviews, we are forced to resort to other methods of demonstrating correctness. Since we cannot tell *a priori* that the design is correct, we must determine this via operational testing. (Note that this does not suggest that security can be "tested in," rather that proper testing can more easily demonstrate a lack of security than can code inspection). The test results of an operational firewall can be considered to be contained within the firewall's audit data. This is true even if you can read (and trust) the source code. Software that contains few or no defects can be installed or executed in an incorrect manner; this is the reason that formal evaluations such as those performed according to [7] are only considered valid for a particular hardware and software configuration, and must be redone for installations that vary from this. [1] presents the case for sustained systems test as one of the means of demonstrating relative correctness; proper collection and examination of audit records on an operational system can be classed as a form of systems test. [8] presents such an approach to Misuse Detection in heterogeneous distributed systems, and suggests that it is effective in detecting intrusions and mitigating risk.

It is fairly easy in practice to audit everything, in volume, to gigabyte disks and helical scan tape. Storage cost per megabyte dropped to almost unmeasurably low levels long ago. Also, given the importance of demonstrating correct implementation, the hardware and software to collect, analyze, and report on network operation is likely to be the most cost effective portion of the security investment. Audit analysis has turned into a well understood art (if not quite a science), and customers will find themselves with a choice of commercial tools available on the market.

A proper audit consists of information concerning all connections (time, length, amount of traffic) and application usage (user IDs, user commands, files accessed). Assuming that sufficient information is included in the audit to uniquely identify all authorized users, it is in theory possible to reconstruct all network data flows after the fact. (For example, by auditing one-time password usage or by cryptographic authentication). This information lends itself to exception scanning and reporting, to alert security personnel to unauthorized access. Note that this will pinpoint not only defects in the security software, but also defects in the implementation (or even in the specification) of access policy.

• *Heresy #4: Firewalling: it's not just for IP any more:* It is cold comfort to people running continent-wide IPX networks (yes, there are indeed such things) that NetWare "doesn't need firewalling." There are many protocol suites besides IP, some of which are widespread enough to require "external" firewalls on multi-access backbones (or even wide area networks): DECnet comes to mind, as does SNA. Unfortunately, current firewalls based on proxy technology only support IP.

An architectural examination of these protocol suites reveals that many of them bear a striking resemblance to IP-based protocols (e.g., NCP/IPX vs. NFS/IP). The argument that an internal firewall should be capable of providing access controls for NFS implies that the firewall should be able to provide controls for NCP as well. The following table provides an example of the commonality of function across several popular protocol suites.

Application	IP	IPX	AppleTalk	DECnet	SNA
Disk Sharing	NFS	NCP	AFP	NETACP	Bulk Data Transfer, Network Data Mover
Routing Information	OSPF, RIP	RIP	RTMP	Hello	n/a
Remote Printing	lpd	NCP	PAP	NETACP	RJE

Table 1: Different Ways to Exchange Data

• *Heresy #5: Robust packet filters can implement robust security:* Routers are good at a number of things. They are fast, support multiple protocols, and have a long history of providing some type of access control using packet filters.

Packet filters have gotten a bad name, mainly because so many implementations have been marginal (to say the least). They have frequently provided minimal filtering functionality (e.g., no source port filtering), insufficient audit capability (e.g., none), and have never tried to filter application data (i.e., no firewalling capability). In addition, most packet filter syntax is opaque (to say the least). The classical discussion on this point is found in [3].

continued on next page

The Firewall Heresies *(continued)*

This is the general case, however, not the theoretical limit, or even the entry point of best practice. [6] presents a packet filtering architecture to address these issues. Consider a router that can filter application data. It could tell the difference between, for example, an NFS read request and an NFS write request. Presumably it could screen out the writes requests but pass the reads. Further, it would do this certainly at Ethernet wire speed (relatively uninteresting), and perhaps at FDDI rates (more interesting).

Given a suitable level of audit capability, it seems that we could construct a multi-protocol firewall out of this device which would provide reasonable assurance of correct implementation as well as high speed performance.

Summary

The point of this entire discussion is that architectures matter, and that we can in fact learn from the past. While early implementations of packet filtering were admittedly inferior, it does not follow that proxy based firewalls are necessarily superior technology to sufficiently robust packet filtering firewalls. Certainly a firewall to protect ATM OC-3 networks cannot be reasonably constructed using a proxy approach. As the world moves to higher speed networks, we should seriously consider alternative architectures that specifically address the performance issue.

The issue of assurance is a complex one, but one which is certainly not being adequately addressed in the community. The persistence of several myths, particularly that source code inspection implies trusted operation, gives customers a misplaced sense of security. While the arguments on analysis of operational systems presented in these pages is abbreviated, it is meant to be a starting point for discussions on true assurance. This analysis is a critical part of any secure system.

Lastly, the issue of firewalling non-IP protocols should be explicitly addressed. Given the large market presence that some of these protocol suites have achieved, it is no longer acceptable to remain IP centric. It is equally unacceptable to require all networks to migrate to IP in order to avail themselves of robust security; in addition to the expense and disruption required by such a transition, some of these technologies provide enhanced services unavailable in IP (particularly in the area of identifying end stations and assigning addresses in a user transparent manner).

References

[1] Marshall D. Abrams, "Belief in Correctness," Proceedings of the 17th National Computer Security Conference; Baltimore, MD, October 1994.

[2] Boris Beizer, *Software System Testing and Quality Assurance,* Van Nostrand Reinhold Company, 1984.

[3] D. Brent Chapman, "Network (In)Security Through IP Packet Filtering," Proceedings of the Third USENIX UNIX Security Symposium, Baltimore, MD, September 1992.

[4] Steven Bellovin and William Cheswick, *Firewalls and Internet Security,* Addison-Wesley, 1994.

[5] Clark, D. D., Jacobson, V., Romkey, J., Salwen, H., "An analysis of TCP processing overhead," *IEEE Communications Magazine,* Volume 27, No. 6, June 1989.

[6] Andrew Molitor, "An Architecture for Advanced Packet Filtering," an unpublished copy of an article to be published in the Proceedings of the Fifth USENIX Security Symposium; Salt Lake City, UT, June 1995.

[7] The U.S. National Computer Security Center, "Trusted Computer System Evaluation Criteria," 1985, DoD 5200.28-STD.

[8] Paul Proctor, "Audit Reduction and Misuse Detection in Heterogeneous Environments: Framework and Application," Proceedings of the Tenth Annual Computer Security Applications Conference, Orlando FL, December 1994.

[9] Dern, D., "Interview with Steve Kent on Internet Security," *ConneXions,* Volume 4, No. 2, February 1990.

[10] Galvin, J., "The Deployment of Privacy Enhanced Mail," *ConneXions,* Volume 5, No. 10, October 1991.

[11] Galvin, J., "Components of OSI: The Security Architecture," *ConneXions,* Volume 4, No. 8, August 1990.

[12] Schiller, J., "Issues in Internet Security," *ConneXions,* Volume 7, No. 9, September 1993.

[13] *ConneXions,* Volume 4, No. 8, August 1990, "Special Issue on Network Management and Network Security."

[14] Galvin, J., "Security Awareness Increasing within the IETF," *ConneXions,* Volume 8, No. 9, September 1994.

[15] Galvin, J., "Summer IETF Meeting Security Report," *ConneXions,* Volume 8, No. 10, October 1994.

[16] Galvin, J., "Security Activities at the Winter IETF Meeting," *ConneXions,* Volume 9, No. 4, April 1995.

[17] Stallings, W., "Cryptographic Algorithms, Part I: Conventional Cryptography," *ConneXions,* Volume 8, No. 9, September 1994.

[18] Stallings, W., "Cryptographic Algorithms, Part II: Public-Key Encryption and Secure Hash Functions," *ConneXions,* Volume 8, No. 10, October 1994.

[19] Stallings, W., "Pretty Good Privacy," *ConneXions,* Volume 8, No. 12, December 1994.

TED DOTY has been involved in Network Security for ten years. He has worked as an developer, designing and implementing network protocols; as a systems engineer, integrating security products into Local and Wide Area networks; and as a programmer, implementing large scale systems of packet filters. He is now program manager for security products at Network Systems, working on that company's next generation of security products. His e-mail address is `ted.doty@network.com`

Problems with the World-Wide Web

by
Mark Handley and Jon Crowcroft, University College London

Introduction

The *World-Wide Web* (WWW) [1] has been a quantum leap for Internet Information servers. From the days of typing obscure incantations full of odd names and numbers, we are now able to get at a plethora of different information at the click of a mouse button. However, there are a number of limitations to the World-Wide Web at the moment, some due to the underlying network, some due to the implementations of clients and servers, and some due to the design of the Web. This article is about some of the problems we perceive. Many of these are being addressed in the Internet Engineering Task Force (IETF) and other working groups.

Technical discussions about the World-Wide Web in general should take place on `www-talk@w3.org`. Discussions about URIs and associated gubbins should take place on `uri@bunyip.com`. Discussions about the HTTP protocol should occur on the Internet mailing list: `http-wg@cuckoo.hplb.hpl.hp.com`. A summary of WWW related mailing lists and archives can be found as:

```
http://www.w3.org/hypertext/WWW/Mailing/Mail/Overview.html
```

Thanks to Simon Spero from the University of North Carolina for this information.

Searching

Hypermedia systems have been primarily designed for browsing. An arbitrary graph of information can always contain a list of links, or an index, which is therefore searchable.

Spiders basically build such indices by walking the Web slowly, a strand at a time (making careful note of where they have been to avoid looping the loop). [2]

Cache hierarchies may make replicated spider-weaving a lot more efficient in terms of the number of retrievals across the net, since the items that don't change and are frequently accessed will be in a server somewhere near each spider's home.

A simple index of the entire WWW would be fairly close to unusable. It is likely that within the foreseeable future, there will be all the books, CDs, and films/videos ever made, available through the Web. There are, for example, 18 million different books in the British Library. A one level index won't work. Thus hierarchical indices and category/key based access systems (Dewey, Library of Congress etc, etc.) will need to be used and some linkage between hierarchy and distribution/replication need to be architected.

Real time

One key problem with the WWW model is how one can add real time media. Currently, the access protocol model is based on the libWWW RPC approach, where a retrieval is initiated by the request, and the client cannot start displaying the result until the return.

Clearly, if the returned data is very large, then this may be inappropriate (see below). However, if the returned data can actually be played as it arrives (e.g., it is audio or video or similar data) then this is clearly not the right way to do things.

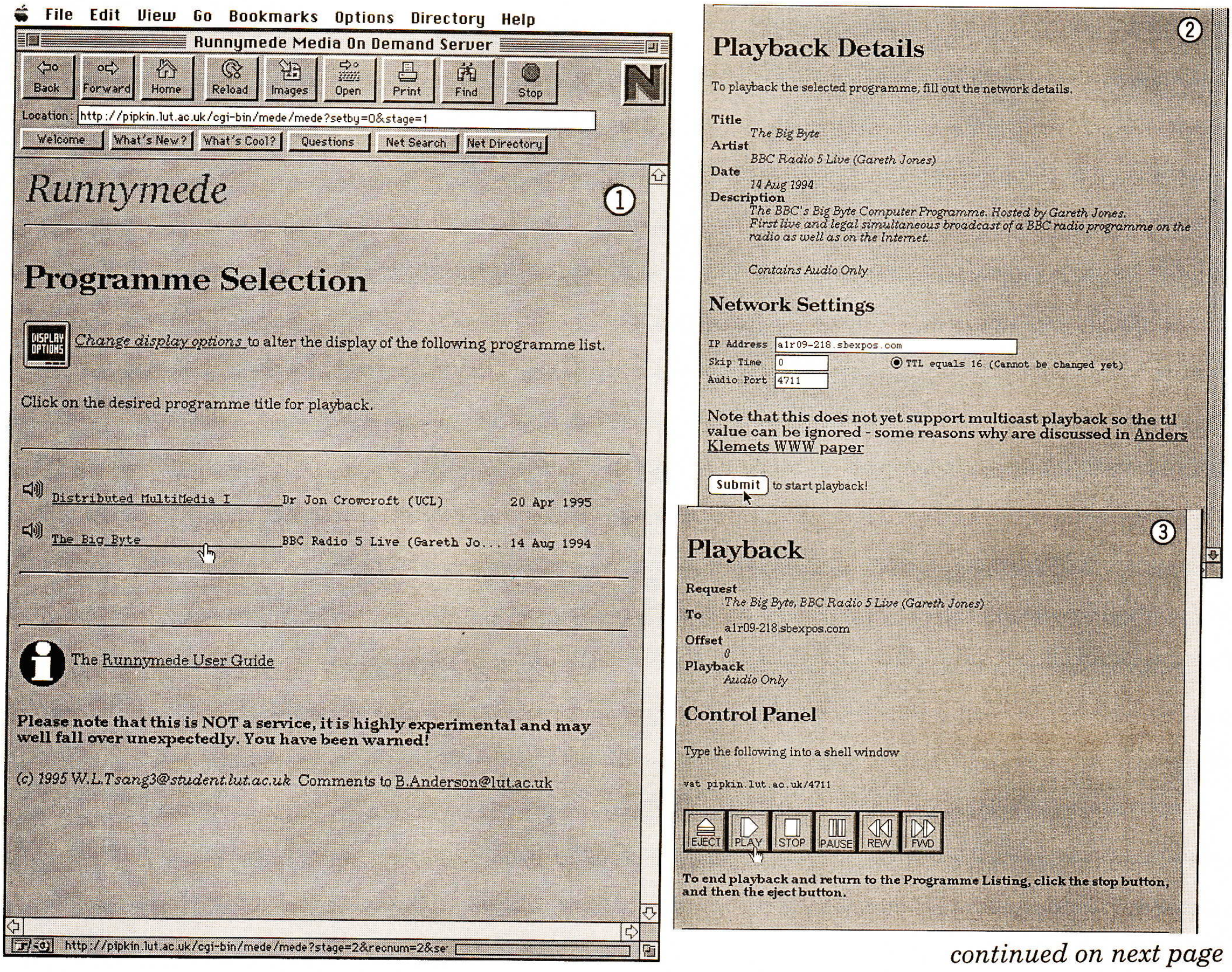
Two approaches suggest themselves:

- Allow a returned stream of data to be played as it arrives as Netscape already does: Client programs launch display programs (or understand the media themselves). Ideally, we would add a new retrieval protocol, for example using RTP, instead of HTTP, to multiprotocol clients.

- Use a separate channel to return real time data.

The latter approach is particularly inviting when one considers some of the ways Internet access may be provided in conjunction with cable: It is quite possible to deliver broadcast quality TV down current copper twisted pair phone wire, up to modest distances, so long as one uses nearly all the bandwidth in one direction. So one could use a second, low bandwidth return channel for HTTP access, and return high quality TV to the client player. One can envisage other networks where bandwidth is asymmetric, or else there are other technical reasons to separate out channels for delivery of data requests and responses from real-time delivery (or transmission). Clients might then be at a multicast address on the Mbone [9] (i.e., one could just run *Wb, Vic, Vat,* etc, and accessing a "seminar-on-demand" URL).

The Runnymede project at Loughborough University of Technology (LUT) has implemented a system based on Anders Klemets Media on Demand [11] server that uses the approaches discussed here. The picture below shows the WWW Interface to the system, which uses *vic, vat, wb* [8] and so on, but does not yet use the Mbone for delivery.



continued on next page

Problems with the World-Wide Web *(continued)*

TCP + RPC problems

In the Internet, much useful information is retrieved from a long way away—that is part of its attraction (it is expensive or slow to get using an alternative material approach).

However, this means that TCP connections between clients and servers traverse long haul networks. TCP is cleverly designed to avoid congestion (network overload). It has a built in conservatism: in the face of packet loss, it assumes that there are other (existing or new) users sharing the network. It therefore backs-off, reducing its sending rate radically (in fact to the minimum effective sending rate conceivable, which is one packet per round-trip-time). It then increases this sending rate until around $^1/_2$ the previous rate which had no loss. It then very slowly increases its sending rate, until it finds what is called a "stable operating point," where a single increase may cause a small problem, but not enough to warrant backing right down again. This approach is called *Congestion Avoidance* and *Slow-Start,* and is known as the "linear-increase, exponential-decrease" approach to traffic control. It was devised by Van Jacobson in the late 1980s in response to real network problems.

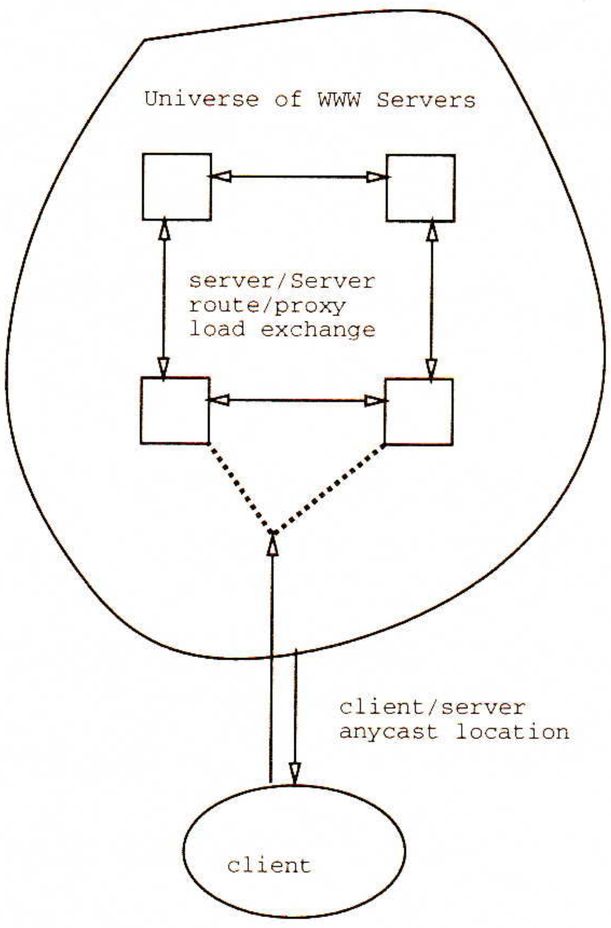
Now, in the absence of any other knowledge, a new TCP connection must start as it means to go on: slowly. This means that when retrieving a page with a large number of separate calls (one for each image), a number of "slow-starts" operate, even when there are no other network users, and there is no apparent reason to operate these. This causes very jerky response to the client program. Unfortunately, this protocol usage is built right through the stack of protocols between the client, the libWWW, and the use of TCP, right through to the semantics of server access.

One way forward would be to not close the connection between the client and server between one HTTP request and the next: since the server end closes the connection, this would entail some extra protocol, at least a marker in the stream from the server to the client to inform it of "End of Response." One could even re-use the multi-body part technology of MIME here. The server should then operate a time-out to prevent clients that die tying up connection resources in the TCP/IP protocol data space.

In fact, a recent modification to TCP [24] specifies an extension that means that TCP can use less packets for short exchanges of messages [as RPC and LibWWW and HTTP tend to], and can cache information concerning the network performance between one connection and a subsequent one to the same destination. This may well solve a lot of these problems when widely deployed. Basically, non-locality of reference by clients will get you every time!

Replicas, caches and consistency

Another problem in the World-Wide Web is also a result of its good design and a cause of its success: each piece of information is held at the server run by the information provider; it is not usually to be found anywhere else. This means that all accesses from no matter which client, no matter where, arrive at the same server. This means that popular servers are hot spots in the network, and cause a lot of network traffic, as well as being potentially overloaded in terms of processing power, by requests. Since the information users and providers in the Internet are quite often totally disjoint from the network providers, there is no particular match between the resources each provide, and popular servers may arise much faster than even the most assiduous network provider can re-dimension their communications links.



Future Web setup including load balanced autoconfiguring protocols.

There are two main solutions for information providers, and third more political one involving the network providers:

• *Provide Caches:* This is already being done. Smart proxy servers can act as clients to other servers. Instead of client programs accessing each and every server around the world, clients always (or usually) access their nearest managed WWW server. It then follows links on their behalf, and keeps copies of the data retrieved for future access by the same or other clients. Such caches must be capable of being invalidated (or out-dated) in some way, if the originator of the information decides to withdraw it. This requires servers to keep track of accesses to them by other servers that are known to cache things, or else some "honesty" mechanism on the part of caching servers, to check whether their cached copy needs updating. (Remember that it will take a lot less time to check the date on an original piece of data than to retrieve it all again, so this could indeed be done every time. The HEAD or GET "if modified after" parts of the HTTP protocol support this). Measurements in 1994 suggest that a single level of caching applied consistently throughout the Internet could reduce traffic by as much as 70%, and increase response items for WWW access and everything else appreciably.

• *Replicas:* Rather than provide "on demand" caches, this approach entails manually tagging data as likely to be popular, and having it replicated across multiple servers when it is installed. Provided a sensible naming system is in place (*Universal Resource Names* [URNs] rather than *Uniform Resource Locators* [URLs]), again clients will go to an appropriately near server, rather than the originator all the time. This is already in use in the world wide archive servers, and has been very effective in reducing network load for FTP traffic. It is, however, more manual than the previous approach.

• *Migration:* If the network providers are prepared to monitor and release statistics of network access patterns to WWW servers, and allow the WWW servers access to topological (link map and bandwidth) information [23] about the network, then it would be possible to migrate information around the network as the center of access patterns from clients became obvious. Again, the naming system must make this feasible transparently to the users.

All three of the mechanisms above require the URN/URI/URL mappings to be in place to work in a completely transparent way, and require some greater degree of intelligence about managing information in the WWW. However, none require centralised management of the Internet or the WWW, so all are still largely in the spirit of the existing successful system. (URI is *Universal Resource Identifier*).

Two interesting facets of replicas and caches to muse upon are charging for resources, and security—If we cache for someone, we are saving someone else network traffic. If we don't provide the same assurances about secure access, we are possibly causing them loss of revenue. Both these problems are complex to solve, although the hooks are there to help provide solutions for all of the problems we have described here, albeit with some extra knowledge on the part of the information managers.

Autoconfiguration

Anycasting is a technique that has been proposed by Craig Partridge and others for service discovery. Basically, anycasting uses multicasting, and a server–server decision protocol to try to achieve a single response to a multicast request in the wide area for "Who Has *x*?" type queries.

continued on next page

Problems with the World-Wide Web *(continued)*

Clearly this is ideal for autoconfiguring clients with proxy/cache servers. It could also be used to resolve URNs to URLs, where there are multiple answers to the question "Who has <URL x>?" The answers would include URCs (*Universal Resource Characteristics*) which may or may not turn out to be application level routing metrics.

But how would a proxy caching server find another? What is needed is a server–server control protocol. Basically, this can be thought of as an application level routing protocol. It needs to exchange messages that describe metrics concerning preferences, which will be based on underlying load, such as network traffic at a server (number of clients), CPU load, cache size and occupancy and so on. The transfer could be based on HTTP, but might be more robust if based on an unreliable protocol. Basically, the problem with using a TCP based protocol to build a distributed application support function like routing, is that TCP masks packet loss and timeouts for some time before reporting errors to the user, whereas by definition, a routing protocol needs timely error reporting to construct a view of which "links" are out of commission. The only time you get a timely outage report from many TCP implementations is when you get an ICMP port unreachable. It is the very percentage of missing responses and delay variation of responses that form part of the metric that is used to choose a "best" server. Thus a UDP based inter-server protocol might be more appropriate.

Ideally, such a scheme would also include metrics gleaned (leaked) from network level routers—for example, delay and link speed can be learned from this (though one would probably avoid including rapidly changing metrics such as queue lengths as these would most likely lead to unstable client access patterns as the traffic and load oscillated due to delays in actions based on learned load).

It has been suggested that future network (IP) level routers might use topology abstraction mechanisms that would permit very efficient client-specific access to such information!

Billing

Charging for information is a very different matter from billing for network use, and will be essential for a number of information providers to offer quality services on the Internet.

The prime requirement is for various degrees of security, and we discuss those in the next section. Currently, we would advise against transmitting billing information or credit card information on the Internet unless you have made yourself very familiar with some of this technology.

The Web makes it quite straightforward to account for access, and therefore to work out a charging scheme. The user is identifiable (subject to security) and each item retrieved is identifiable, and most servers already provide extensive logging (but see "caching"). It is then simply a matter of working out a policy, and implementing it. Note that without a guaranteed service delivery channel it is hard to bill for timeliness of delivery of information.

Distributed execution

The World-Wide Web has a client–server architecture. Humans initiate accesses at the client and the server, or its proxy, responds with the Web page or a cached copy. The *Common Client Interface* (CCI) allows programs to drive the client (though how they do it differs currently between different client implementations).

The *Common Gateway Interface* (CGI) allows servers to be driven by programs/scripts that are executed as a result of access to the page that "names" the script. CCI and CGI both allow standardised forms of parameter passing from these external programs to the clients and servers.

One client implementation, Netscape 1.1, has added facilities for *Client Pull* and *Server Push,* which allow input other than a human's to start a chain of events. Server Push is simple: A server script produces a multipart MIME message rather than the normal single body part. The client renders each MIME part as it arrives, and retains the underlying HTTP/TCP connection, awaiting the next body part. Each has its own MIME content-type. Client Pull is achieved by using META tags to include pieces of HTTP in HTML. In other words, the document being accessed includes HTTP, so that as well as being rendered by the client, it causes the client to take HTTP protocol actions; for example, a directive to "Refresh" the page after some delay causes the client to get the page again. If the document is a script, this can cause successive responses to be different.

However, there is no current way to build an autonomous set of programs that "live" in the Web. Ongoing work at UCL is addressing this through a general purpose programming system called *LAW.* Tim Berners-Lee refers to WWW as the DNA of the Internet. LAW is a *Language for Agents in the Web.*

For a long time, people have been talking about active objects, and about agents. Scripting languages have been developed that might enable these type of things to be built, but the run-time system for such languages has been too centralised. Well, now we have 2 pieces of technology that enable us to build agents properly: The WWW and the Mbone (with RSVP and *Class Based Queueing* [CBQ] and the *Conference Control Coordination Protocol* [CCCP]).

Names, Objects and Types

LAW is an idea for a safe language where HTML is a subtype, but URLs are first class data types. A URL is the "address" or "reference" type familiar to assembler or **C** programmers (note it is *not,* explicitly not a URN—A URN is location independent—URNs might be provided through a system written in LAW). Finally, a new MIME type is added so that clients know that what is returned from a server is a piece of the LAW.

CCCP names are hierarchical names that can include DNS names. The LAW extends these to include URLs, and URLs to include these.

We also name other data types, such as media types simply by using MIME types. All other objects, as in *Tcl* or *Telescript,* are strings. This is a predefined type merely for convenience and efficiency (like stdio/iostream). [A Touch of SNOBOL or BLISS!]. There are a lot of other languages to look at for example *Linda,* and other agent work, and *Java*. [14] Java is **C++** like. We believe that this is possibly a misguided design choice. The authors say that "most programmers are used to **C++**." Sure, but most Web users are not programmers. See also the *Mobisaic* [15] work at Washington for some similarly neat ideas on agents and the Web.

However, LAW does not define non-string operators on these types without the programmer having to go through all the hoops (hops) of a full *Abstract Data Typing* activity. Thus it is type safe and type extensible. Scoping of types is yet to be devised, but certainly, containment hierarchies (as in network management) seem like a good approach.

continued on next page

Problems with the World-Wide Web *(continued)*

Persistence is simply given by writing new Web pages. A new HTML tag, `<law SomeLawProgram>`, allows embedding of laws in the Web.

LAW is interpreted, and is readable, but can be incrementally compiled (has an efficient representation, is codified).

LAW is functional. There exists good reference material on functional language implementation and also, in particular, the very fine *gofer*. LAW programs are stateless, so that they are fault tolerant and can be re-executed harmlessly (idempotency). LAW programs output can be stored in the Web. It remains to be seen whether filters (lazy evaluators of event streams) may wish to store their existence persistently in the Web.

Communication

Agents of the LAW are run by WWW servers through messages arriving that indicate a script to run through CGI. Initiators and responders for users either receive passive objects (pages) in HTML, or else receive a new MIME type, application/act, which is passed to a User Agent, launched as a "viewer." This can continue communication (i.e., save state) through CCI back to the Web client, or might receive Legal Data back in the script, that gives it a communications handle direct back to the server.

LAW has communications primitives similar to UNIX UDP sockets (or to *Tcl* or *Telescript*): `send()` and `recv()` are basic ways to direct a message to another Agent of the LAW. Multicast is allowed.

Laws may be passed that are qualified by a traffic class [17] that permits them to be prioritised in their application. This has clear uses in networks that provide resource control, in accordance with the scalable approaches (that permit aggregation of resource classes) such as CSZ and CBQ. This will determine how many Laws can be passed by a server or client, or from a server to a client and so forth.

Who controls the Agents of the LAW?

Agents follow control structures exactly as in *Tcl* (why give up a good thing). Just as in *Tcl,* the LAW can be extended.

Implementation of the LAW

Client and servers currently interact with the LAW through CCI and CGI. Obviously as with other approaches based on WWW, we'll eventually construct clients (and servers) entirely within the LAW, as proof of concept, but also for additional security. An interim approach is to add a LAW evaluator with an interface to *Tk,* as part of a Web client.

Applications in pursuit of the LAW

One obvious application of the LAW is the construction of URRs. URRs are *Uniform Resource Routes.* There are two types of URR:

- URRs for exchange of metrics between proxy caching servers and clients: These are used to load balance—i.e., the basic idea that a URN lookup returns a list of URLs is a starting point, but how does a client choose between different URLs for the same URN? Easy—by looking at URRs.
- URRs that allow you to track the correct semantic result of a URN lookup: These are much more like agents, that look at user preferences and all kinds of similar stuff.

You can see that the first of these is low level, while the second is high level, but that they can conspire together to mutual benefit.

Another application is in support of Webcasting. Webcasting is the use of the Mbone and synchronised clients to allow distance learning in the same way as *wb,* but possibly without the data distribution problems.

The use of LAW to aid in Webcasting is simple:

You need to pre-load the caches near all the clients. You can use multicast to do this. In fact, at the start of a Webcast, you can start the Webcast multicast to all the clients cache servers (helped in the process of finding them of course by inverse URR lookups), and just continue (UNIX file block loading style—if someone is going to read a byte of a file, there is a good chance that they will read all of it. In the Web, this is not generally true, but in a Webcast it is). Then the protocol is simply one of using remote control Netscape or CCI (perhaps through a common LAW interface) to step each person through to the right page on their locally cached (at proxy/server, or in some cases, right at the client) copy. Repairs to the reliable initial cache load are done *wb* style (i.e., cache servers multicast requests for missing pages, and the "nearest and least dearest" answer them).

We have to stop thinking about applications for the LAW, as every time we start, we come up with 100 more, and don't have time to finish the LAW, and start on the legislation :–)

Security

Security concerns are typically broken down into areas by thinking about attacks. For example, you might think that if a site *S* wants to serve a client *C*, with Web page *W*, it is sufficient that:

- There is authentication of the identities *S* to *C* and *C* to *S*
- There is authentication of the contents *W* from *S* to *C* (or *C* to *S* for posting a form)
- There is non-disclosure of *W* to non-*C*
- Non-disclosure of the identity of the *C, S, W* triple to any other *C* or *S* (anonymity)

There is a refinement of this—it may be of interest to third parties that *C* is accessing any *W* on *S*—i.e., that *C* and *S* are in cahoots (e.g., competitors/takeover bids, traffic analysis etc.):

- It may be necessary to identify that some *C* has communicated with *S,* but to disassociate that fact from the contents (e.g., right not to incriminate self)
- It is possible to use sideband information in HTTP to do covert signaling
- It is relatively easy to do a PCB attack on TCP and deny service at a server *S*

PGP [10, 25] (and similar technology being deployed soon in clients and servers) only solve the first four of these conditions, usually.

Privacy

Under this heading, basically, we are talking about non-disclosure of information except to those authorised (typically creator and anyone they wish to transmit it to, subject to proof of identity!) If I send something over a communications channel, it is liable to be eavesdropped, copied, intercepted in many different ways. The solution is to mangle (encrypt) what is sent so that someone who takes a copy can't recognise it. The easy way to do this is to have "one-time-codebooks" (e.g., each letter is sent as a number which is its next occurrence in some pre-agreed book). This is inefficient, complex, and error prone, and not a lot of use for WWW servers. A more useful technology is cryptography, and in particular, a mechanism called public key cryptography, of which more below. A less useful (but still effective) technique is private key cryptography, (such as the *Data Encryption Standard,* DES).

continued on next page

Problems with the World-Wide Web *(continued)*

All cryptographic technology for privacy is export controlled from the US at the moment, which means that use every day in the Internet relies on externally produced implementations. Also, for International companies trading in the US, this is a political obstacle to using such technology for fear of offending the US government.

In fact, for many WWW servers, privacy may not really be an issue—especially if the function of the server is effectively to add value to other services (advertise). Even if not, it may simply be too difficult for someone to monitor all WWW traffic from a particular server and piece together all the data from there—but do not count on this one bit!

Authentication

This is to do with proving ones identity. It is a very subtle business. Typically, it relies on some notion of trust. If I transmit something to someone by talking to them face to face, I am assured by their face, or voice, that they are who they say they are. However, I need to also understand their role—do they really work for who they say they do? Typically, they produce credentials (an expensive to fake ID-Card, for example).

If I send something over a communications channel, I must also exchange some forms of credentials to be assured the receiver is who they say they are (like listening to their voice or looking at their face, I may need a notarised signature etc, etc). There are a number of techniques for digital signatures that are hard to forge. Luckily, this technology is also not subject to US export controls. Systems such as Kerberos [25] and PGP (*Pretty Good Privacy*) [26] provide authentication, as well as possibly non-exportable privacy.

Once you have authentication, then a server can be protected from dangerous access. It can then match authenticated credentials to Access Control Lists, and even carry out billing based on these (though it still better not accept credit card numbers or send bills over the same network until it also has privacy technology).

Non-repudiation

Non-repudiation of transmission/reception is the facility of a secure system to permit proofs that a sender or receiver were indeed the sender and receiver. Non-repudiation of contents is to do with being able to say that although a party communicated, they refuse to say whether the contents that some other party reveals are in fact the contents of the aforesaid communication [shades of "Yes, Minister"].

Integrity

This is to do with making sure that information is not tampered with in transit. It is usually achieved by signing the data with some function (checksum) of the data itself, wrapped up with some key that is not transmitted (perhaps has been exchanged previously through public key cryptography).

We might add another: *legal recognition!* All the security in the world is not much use if it is not backed up by some sort of legal position, if you want to charge people for information, exchange contracts over networks etc etc Very very paranoid sites are concerned with two more security facets: Traffic pattern analysis and covert signaling. Traffic pattern analysis might concern finance houses who would be worried if it was known which information providers they were gathering information from, for example, since it might permit others to trade on this knowledge (impending mergers etc). Covert signaling is a way of carrying information piggy-backed on legitimate information. This is very handy for spies getting information out of secure sites.

Finally, note that keeping logs is a very important part of security. Any secure system will attract more attempts to hack simply because it poses more of a challenge, but also because there is potentially more monetary gain hacking a system that someone has apparently spent money to protect. A system always has vulnerabilities (there is no such thing as 100% security, just 100% lack of detected hacks). If you keep audit trails, then it is possible to track a lot more problems. It may also be possible to track the cause of the loophole and fix it.

Public Key Cryptography and the WWW

Public Key Cryptography (PKC) is a very neat technology. Private key systems involve people sharing a single private key and each person having to give each trusted person a copy of their private key via some secure channel—if they have this secure channel, why not just use it anyhow for all communications (actually it may be a more costly channel, but never mind...).

The idea is that instead of one key, there are two. A private and a public key. When I create a key, I actually create these two interrelated keys. If I encrypt things with my private key, it is decryptable with my public key but not with my private key and vice versa. How is this useful? Well I now no longer have a key distribution problem that I had for private keys (needing some magic special secure channel), since I can publish my public key (e.g., in the WWW on my home page). In fact, there is an interaction between caches and privacy which may be protected by the use of secret and public key cryptography, whereby we get a SK from some origin by PKC, and the data from the cache.

The popular mail authentication and security package PGP uses such technology, and promises to be directly applicable to the WWW.

Netscape 1.1 now offers RSA PKC as a means of providing this functionality. A minor nit is that the US version has a 128-bit key, while the export version has a 48-bit key which is inadequate for most serious commercial ventures (e.g., banks, who already have export control exemption for DES).

A combination of symmetric (private) and asymmetric (public) key cryptography could be usefully combined with proxy/caching servers to provide scalable, heterogeneous security, but that is a topic for another article.

Infrastructure changes that will help

The WWW community is currently working on new versions of HTTP, HTML and the whole architecture of URN, URI, URL, URC and maybe URRs.

The program language community has developed safe interpreted languages such as *perl, python,* and *tcl* in the public domain, as well as *Java* and *Telescript.*

We live in exciting times!

References

Items marked "Internet Draft" are "Work in Progress" of the IETF and are available from several depositories on the Internet. The appropriate draft will depend on the time of reading rather than writing of this article, so we have omitted specific URNs.

[1] Crowcroft, Jon and Handley, Mark "The World-Wide Web: How Servers Work," *ConneXions,* Volume 9, No. 2, February 1995.

[2] Koster, Martijn, "Robots in the Web: threat or treat?" *ConneXions,* Volume 9, No. 4, April, 1995.

continued on next page

Problems with the World-Wide Web *(continued)*

[3] C. Partridge, T. Mendez, W. Milliken, "Host Anycasting Service," RFC 1546, November 1993.

[4] T. Berners-Lee, L. Masinter, M. McCahill, "Uniform Resource Locators (URL)," RFC 1738, December 1994.

[5] N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies," RFC 1521, September 1993

[6] K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text," RFC 1522, September 1993.

[7] S. Deering, "Host Extensions for IP Multicasting," RFC 1112, August 1989.

[8] S. Casner, S. Deering, "First IETF Internet Audiocast," *ConneXions,* Volume 6, No. 6, June 1992.

[9] M. Macedonia, D. Brutzman, "MBone Provides Audio and Video Across the Internet," *IEEE Computer,* April 1994.

[10] Stallings, W., "Back to Basics: Cryptographic Algorithms, Part I: Conventional Cryptography," *Connexions,* Volume 8, No. 9, September 1994.

[11] Anders Klemets, "The Design and Implementation of a Media on Demand System for WWW," Department of Teleinformatics, The Royal Institute of Technology, Stockholm, Sweden.
`http://www.it.kth.se/~klemets/www.html`

[12] Linda information and papers can be found at:
`http://www.cs.yale.edu/HTML/YALE/CS/Linda/linda.html`

[13] Agents are discussed a lot at:
`http://www.smli.com/research/tcl/lists/agents-list.html`

[14] Java is available from Sun at:
`http://java.sun.com/`

[15] Mobisaic papers can be found at:
`http://www.cs.washington.edu/homes/voelker/mobisaic/mobisaic.ht`

[16] Gofer is a functional language and implementation with a highly efficient compiler:
`http://www.cs.nott.ac.uk:80/Department/Staff/mpj/`

[17] Wakeman, Ghosh, Crowcroft, Jacobson and Floyd, "Implementing Real Time Packet Forwarding Policies using Streams," Proceedings of USENIX Conference, New Orleans, January 1995.

[18] Bob Braden and Lixia Zhang, "RSVP: A Resource ReSerVation Protocol," *ConneXions,* Volume 8, No. 8, August 1994.

[19] Schulzrinne/Casner/Frederick/Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF Audio-Video Transport Working Group, March 1995, Internet Draft.

[20] T. Berners-Lee, R. T .Fielding, H. Frystyk Nielsen, "Hypertext Transfer Protocol—HTTP/1.0," IETF HTTP Working Group, March 1995, Internet Draft.

[21] Dave Raggett, "HyperText Markup Language Specification Version 3.0," W3C, March 1995, Internet Draft.

[22] Paul E. Hoffman, Ron Daniel, Jr., "IETF Generic URN Syntax," IETF URI Working Group, Internet Draft.

[23] I. Castineyra, J. N. Chiappa, "The Nimrod Routing Architecture," March 1995, Internet Draft.

[24] R. Braden."T/TCP—TCP Extensions for Transactions Functional Specification," RFC 1644, July 1994.

[25] B. Clifford Neuman and Theodore Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications,* Volume 32, No. 9, September 1994.

[26] Stallings, W., "Pretty Good Privacy," *Connexions,* Volume 8, No. 12, December 1994.

[27] Schiller, J., "Issues in Internet Security," *ConneXions,* Volume 7, No. 9, September 1993.

[28] *ConneXions,* Volume 4, No. 8, August 1990, "Special Issue on Network Management and Network Security."

[29] Schiller, J., "Kerberos: Network Authentication, *ConneXions,* Volume 4, No. 1, January 1990.

[30] Kaliski, B., "An Overview of Public-Key Cryptography Standards," *ConneXions,* Volume 6, No. 5, May 1992.

[31] Stallings, W., "Back to Basics: Cryptographic Algorithms, Part II: Public-Key Encryption and Secure Hash Functions," *ConneXions,* Volume 8, No. 10, October 1994.

[32] Berners-Lee, T., "A Summary of the WorldWideWeb System," *ConneXions,* Volume 6, No. 7, July 1992.

[33] Berners-Lee, T., R. Cailliau, A. Loutonen, H. F. Nielsen and A. Secret, "The World-Wide Web," *Communications of the ACM,* Volume 37, No. 8, August 1994.

[34] Luotonen, A., K. Altis, "World-Wide Web Proxies," Proceedings of the First International World-Wide Web Conference, May 1994.

JON CROWCROFT is a senior lecturer in the Department of Computer Science, University College London, where he is responsible for a number of European and US funded research projects in Multi-media Communications. He has been working in these areas for over 14 years. He graduated in Physics from Trinity College, Cambridge University in 1979, and gained his MSc in Computing in 1981, and PhD in 1993. He is a member of the ACM, the British Computer Society and the IEE. He was general chair for the ACM SIGCOMM 94 symposium. He is also on the editorial teams for the *Transactions on Networks* and the *Journal of Internetworking.* E-mail: `J.Crowcroft@cs.ucl.ac.uk`

MARK HANDLEY graduated from UCL with a 1st class honours degree in Computer Science with Electrical Engineering in 1988. As a PhD student at UCL, he studied novel neural network models and their visualisation. Since 1992, he has been a Research Fellow, working on the RACE CAR project and on MICE, now managing the UCL part of MICE. His current research interests include Multimedia Systems, especially audio and video encoding and compression, Distributed and Heterogeneous Systems, and HCI and graphics. E-mail: `M.Handley@cs.ucl.ac.uk`

Letters to the Editor

Dear Ole,

Java

Well, things sure move fast in the Net. Since sending that article to you on World-Wide Web Futures (a dangerous thing to write, I know!), the ideas I wrote about in LAW, have been overtaken by real world events. Sun Microsystems have licenses their language, *Java* to Netscape Communications Corporation. Java is an object oriented programming language, so we get three trendy ideas in one place—object orientedness, the Web and agents.

C++ like

One might have expected Sun to have come out with something perhaps based on *Tcl/Tk* (and maybe one of its object oriented offshoots), but no, instead, a whole new **C++** like language has been developed. The idea is that **C++** is all the rage, so many programmers know it already. By slimming it down, and adding a few neat things such as auto-garbage collection, true object-oriented interfaces (dynamic inheritance), distribution (via HTTP/FTP) and some ideas imported from the safe-tcl/safe-perl/safe-python communities, Java leverages off a lot of well known technology...

Of course, many of those so "taken" with the Web are not programmers, and it remains to be seen if such a powerful language is right for the average publisher, architect, civil servant or pizza shop employee, but we sure do live in exciting times.

Cheers,

—Jon Crowcroft, University College London
`J.Crowcroft@cs.ucl.ac.uk`

P.S. The URL is `http://java.sun.com/` (of course)

Dear Ole,

In "History of the Internet" by John S. Quarterman in the April 1995 issue of *ConneXions* there is no mention of EuropaNET or DANTE, which makes the history of the (European) Internet incomplete. I would like to take this opportunity to add some dates/events.

Best regards,

—Josefien Bersee DANTE External Relations Manager

P.S. DANTE's Web server is: `http://www.dante.net`

Additional timeline entries

October 1992: EuropaNET, pan-European Multi-Protocol (partly 2 Mbps) backbone is launched, connecting national research networks in 18 European countries. EuropaNET has AUP attached to it, network for research-related traffic only.

July 1993: DANTE is established by eight national and one regional (5 Nordic countries) research networks in Europe. Main objective: manage and develop EuropaNET for and on behalf of national networks.

1993: 18 European countries and EC come together in EuroCAIRN Project. Objective: establishing a pan-European 34 Mbps and higher speed infrastructure for research as soon as possible.

May 1994: DANTE is contracted by EuroCAIRN to carry out survey of requirements, and produce technical and commercial implementation plan.

October 1994: AUP on EuropaNET is dropped for practical reasons, no real changes occur. Research networks' interests continue to dictate development of the network.

March 1995: DANTE's report to EuroCAIRN, blueprint for next generation high speed pan-European research network, is approved by EuroCAIRN committee.

DANTE submits project proposal (as coordinating partner in TEN-34 consortium, see below) for funding of Interconnection of high speed national research networks under 4th Framework Programme of the EC.

April 1995: New EuropaNET contract is awarded to BT—network with nodes in 15 countries up to speeds of 8 Mbps will be established. No AUP attached, open to commercial customers.

May 1995: Establishment of TEN-34 consortium formally announced, 18 European countries with DANTE as coordinating partner participate in project proposal to EC Fourth Framework Programme (see above) to set up a high speed pan-European backbone for research. TEN-34 proposal receives favourable evaluation report.

The Author responds

Dear Ole,

I was pleased to see the message from Josefien Bersee about EuropaNET and DANTE; it contains just the sort of information I was hoping the history article would elicit. I encourage others to send in their items, as well. I'd be happy to provide an updated timeline later for *ConneXions,* and we'll also be publishing updates in the MIDS monthly newsletter *Matrix News,* and in the second edition of the book, *The Matrix.*

Thanks,

—*John S. Quarterman,*
Matrix Information and Directory Services, Inc.
`jsq@tic.com`

Write to *ConneXions*!

We'd love to hear your comments, suggestions and questions about anything you read in *ConneXions.* Our editorial address is given below. Use it for letters to the Editor, requests for the index of back issues, questions about particular articles etc.:

ConneXions—The Interoperability Report
303 Vintage Park Drive
Suite 201
Foster City
California 94404–1138
USA
Phone: +1 415-578-6900 or 1-800-INTEROP (Toll-free in the USA)
Fax: +1 415-525-0194
E-mail: `connexions@interop.com`
URL: `http://www.interop.com`

Subscription information

For questions about your subscription please call our customer service hotline: 1-800-575-5717 or +1 502-493-3217 outside the USA. This is the number for our subscription agency, the Cobb Group. Their fax number is +1 502-491-8050. The mailing address for subscription payments is: P.O. Box 35840, Louisville, KY 40232–9496.

Book Reviews

Routing in Communications Networks, by Martha Steenstrup, (Editor), Prentice Hall, 1995, ISBN 0-13-010752-2 and *Routing in the Internet,* by Christian Huitema, Prentice Hall, 1995, ISBN 0-13-132192-7.

Background

Routing and routing protocols are mysterious topics to most people, including many networking professionals. Although numerous books discussing communications protocols and the Internet have been written in the past couple of years, there has been a noticeable gap in coverage in the area of routing. While most comprehensive communications books discuss routing in the obligatory one or two chapters, until this year there were only a couple of books available that focused on routing protocols and internetworking (bridging and routing) technologies. The only other choices for information on these topics were academic papers, journal articles, RFCs, and other standards documents. Two new books on routing and routing protocols became available this spring, helping to fill this gap.

Steenstrup

Routing in Communications Networks by Martha Steenstrup consists of eleven chapters written by different authors. The chapters are arranged into four parts: Circuit-Switched Networks, Packet-Switched Networks, High-Speed Networks, and Mobile Networks. Steenstrup provides several pages of introduction to routing and routing protocol issues framing the material for the entire book. Each of the four parts also includes an overview of the concepts to be presented and pointers to material which is not addressed directly in the chapters.

Both chapters in the first part, "Circuit-Switching Networks," introduce the major challenges for this technology as call (virtual circuit) acceptance and call (virtual circuit) routing. *Dynamic Alternative Routing* (DAR), a technique used for call setup in telephone networks is discussed in the first chapter. The other chapter in the first part covers the *Least Loaded Routing* (LLR) algorithm for ATM networks. An overview of ATM is provided along with the challenges encountered in developing routing strategies for ATM networks as contracted to telephone networks. The details of three variations on the LLR algorithm are presented as solutions to these challenges. Unfortunately, there is very little discussion of alternative algorithms.

In the "Packet-Switched Networks" part of the book, the chapter on distance-vector routing introduces concepts such as split horizon, discusses update and convergence issues, and then contrasts different implementations of distance-vector protocols. The inter-domain routing chapter includes information about the *Exterior Gateway Protocol* (EGP), *Border Gateway Protocol* (BGP), and *Inter-domain Routing Protocol* (IDRP), presented in the context of evolution of the Internet. The chapter on link-state routing is very similar to the distance-vector chapter in form, that is, a discussion of link-state concepts is followed by information about specific implementations in several protocols. AppleTalk routing is the final chapter dealing with packet-switching. This chapter includes information about several different protocols from the AppleTalk suite including RTMP, ZIP, and NBP, but most of the emphasis of this chapter is on AURP, the *AppleTalk Update-based Routing Protocol,* and how it can be used to improve AppleTalk routing in large environments. Each of these chapters in this part is excellent, but there is little explanation of how the topics relate to each other.

The "High-Speed Networks" part of the book lists motivations and challenges applicable to evolving high-speed network technologies. This part includes a comprehensive survey of routing in optical networks, an introduction to packet-level and call-level routing in the plaNET system, and a chapter on deflection routing which focuses on performance analysis of this alternative to traditional packet forwarding. This part of the book provides visibility into a rapidly developing area. The three chapters together are a fine introduction to some of the issues that are faced as networks get faster.

The final part, "Mobile Networks," includes two chapters. Each chapter is a technology survey, the first being routing in cellular mobile radio networks, the second is packet radio routing. The chapter on cellular includes material on CDMA, TDMA, and CDPD network technologies and architectures. The packet radio chapter uses the DARPA SURAN project as an example to explain the challenges and techniques of this technology. While it is difficult to provide comprehensive coverage of such a broad topic as mobile networking, the brief pointer to IETF standards activity in this area and only a few paragraphs on satellite applications are a disappointment.

A great feature of this book is the extensive bibliography associated with each chapter. It is great to see such a varied survey of material become available. The only problems with this book are that we need more material and in some cases, better integration.

Huitema

Routing in the Internet by Christian Huitema is more focused, as you might expect, on the routing and related protocols in the TCP/IP protocol suite used in the Internet. The book consists of an introduction to the Internet architecture and IP protocol suite, including justification for the primary architectural decisions, and discussions and comparisons of interior routing protocols and exterior routing protocols. The final part of the book covers areas which have received a lot of attention in the Internet community recently, specifically multicasting, mobility, and resource reservation.

Huitema details RIP and RIPv2 as examples of distance vector protocols, and OSPF as the recommended link state protocol for the Internet. The protocols are compared from a functionality and complexity perspective. Information on IS–IS, IGRP, and Enhanced IGRP is also provided and compared. These chapters are a fine overview of interior routing protocols, however the benefits described for link state technology blur the distinction between technology differences and actual protocol implementations. There is an error in the section on IGRP relating to variance, it is still settable in latest versions. Unfortunately, considering its widespread use, the best description of default routing is hidden in the section on IGRP rather than being expanded and set apart. The ASCII T-shirt drawing provided a fun recollection of the routing protocol wars of the early 1990s.

The part on Exterior Routing Protocols is developed chronologically, covering the details and limitations of EGP and its resulting displacement by BGP. The BGP protocol is described, along with important issues of interaction with interior routing protocols. *Classless Inter-Domain Routing* (CIDR) is described along with other new developments in inter-domain routing, discussing the challenges of explosive Internet growth and possible futures.

continued on next page

Book Reviews *(continued)*

Chapters on three emerging technologies, multicast routing, IP mobility, and resource reservation, make up the final chapter of the book along with some brief comments on the next generation of IP.

Comparison and Summary

While both books include material on mobility, and inter-domain, distance-vector, and link-state routing, the different perspectives provide different views of the technologies. The Steenstrup book addresses a much broader technology base, presumably appealing to a wider audience including researchers. The Huitema book is more focused, addressing the needs of the network manager actually using IP routing protocols and connecting to the Internet.

Both books do an admirable job of attempting to describe the state of the art in routing, but various details have changed in several areas, even during the few months since the books were written. In such a rapidly developing field, however, it is inevitable that printed books will be at least somewhat out of date. Rapid developments in the technology are the same reason that these books are both so valuable in providing overviews to the field of routing and routing protocols.

—David O'Leary, Cisco Systems
doleary@cisco.com

Future NetWorld+Interop Dates and Locations

NetWorld+Interop 95	Tokyo, Japan	July 17–21, 1995
NetWorld+Interop 95	Paris, France	September 11–15, 1995
NetWorld+Interop 95	Atlanta, GA	September 25–29, 1995
NetWorld+Interop 96	Las Vegas, NV	April 1–5, 1996
NetWorld+Interop 96	Frankfurt, Germany	June 10–14, 1996
NetWorld+Interop 96	Tokyo, Japan	July 15–19, 1996
NetWorld+Interop 96	Atlanta, GA	September 16–20, 1996
NetWorld+Interop 96	Paris, France	September 23–27, 1996

All dates are subject to change.

Call 1-800-INTEROP or 1-415-578-6900 for more information. Or send e-mail to info@interop.com or fax to 1-415-525-0194. For the latest information about NetWorld+Interop including *N+I Online!* as well as other SOFTBANK produced events, check our home page at http://www.interop.com

NetWorld+Interop is produced by SOFTBANK Exposition and Conference Company, 303 Vintage Park Drive, Foster City, California 94404–1138, USA.

Announcement and Call for Papers

Topics

The IFIP conference *Performance '96* will be held October 9–11, 1996 in Lausanne, Switzerland. Performance '96 is an International Conference on Performance Theory, Measurement and Evaluation of Computer and Communication Systems. It provides a forum where theorists and practitioners from a variety of application fields present state-of-the-art work and exchange new ideas. Topics include:

— *Performance-oriented design and Evaluation studies of:*

- High-speed networks, ATM
- Protocol modeling
- Router networks
- Network management/control
- Multimedia systems
- Mobile communication
- Client–server computing
- Distributed/parallel systems
- File and I/O systems
- Database systems
- Operating systems
- Computer architecture
- Real-time systems

— *Techniques and algorithms for:*

- Performance optimization
- Workload characterization
- System measurement
- Simulation
- Statistical analysis
- Reliability
- Stochastic modeling
- Queueing analysis
- Performability analysis
- Scheduling theory
- Timed/stochastic Petri Nets
- State-space methods

Submissions

You are invited to submit papers, proposals for parallel sessions or tutorials. Papers should report original and unpublished research. Manuscripts should not exceed 20 double-spaced pages, excluding figures and tables. Electronic submission is encouraged. Manuscripts in plain ASCII or *PostScript* formats can be submitted via e-mail to: **`perf96-submit@lrc.epfl.ch`**

It is also possible to submit a paper version. In that case, send 9 copies to the program committee chairman. Author identification should appear on a separate cover sheet and *not* within the paper itself. In all cases, please send e-mail to **`perf96-submit@lrc.epfl.ch`** giving address of contact person, authors, title and abstract.

Important dates

March 15, 1996:	Submissions due
June 10, 1996:	Author notification
July 15, 1996:	Final revised manuscript due

Publication

Approximately 20 fully refereed papers will be presented in the main track of the conference and appear as a special issue of *Performance Evaluation.* Papers being presented in parallel sessions will be made available through the World-Wide Web.

Contact addresses

Jean-Yves Le Boudec, General Chair
Laboratoire Réseaux de Communication
Swiss Federal Inst. of Technology
CH-1015 Lausanne
SWITZERLAND
`leboudec@di.epfl.ch`

Martin Reiser, PC Chair
ETHZ/IBM
Zürich Research Lab.
Säumerstraße 4
CH-8803 Rüschlikon
SWITZERLAND
`reiser@inf.ethz.ch`

Call for Papers

The *Second International Workshop on High Performance Protocol Architectures,* HIPPARCH '95 will be held in Sydney, Australia, December 11–12, 1995. This is a workshop organised by University of Technology, Sydney within the context of an Australian European Collaboration project sponsored by CEC DG XIII and the Australian Bilateral Science and Technology Program.

Objective and scope

The aim of the HIPPARCH workshop is to evaluate high performance techniques for the implementation of communication subsystems, especially the use of "Application Level Framing" (ALF) and "Integrated Layer Processing" (ILP) concepts. It will also be used to present the results of the HIPPARCH project, and will thus provide an excellent environment for dissemination of information for researchers working in this area.

Topics

Topics of interest for which original research papers are solicited include:

- Adaptable transmission control mechanisms
- Implementation techniques
- Experiences with ALF/ILP
- Tools and description languages for protocol implementation

In order to maximise the benefits of a workshop of this nature, we strongly encourage submission of papers which describe on-going research and of implementation experiences.

Submissions

Extended abstract of approximately 1500 words plus position statement (including references to the current research in the field) may either be submitted by electronic mail, in *PostScript* format to:

`hipparch-workshop@ee.uts.edu.au`

or

HIPPARCH Workshop Secretary
School of Electrical Engineering
University of Technology, Sydney
PO Box 123
Broadway NSW 2007
AUSTRALIA

Publication

Selected papers from the Workshop will be invited to be submitted to the *Australian Computer Journal.*

Important dates

Abstracts due:	15 October, 1995
Acceptance Notification:	15 November, 1995
Final Paper Submission:	1 December, 1995

Venue

The workshop will be held at the Markets Campus of the University of Technology, Sydney. This lies at the southern end of Sydney's Central Business District. It is also adjacent to Darling Harbour, which is recognised as one of the major urban renewal projects of the last decade. Darling Harbour consists of parks, shopping malls and entertainment areas, as well as hotels.

Sydney

Contemporary Sydney was established when the first European settlers landed at Sydney Cove on the 26th of January 1788. Since then it has grown to be Australia's largest city. It is the gateway to Australia, serviced by daily flights from Europe, USA and East Asia.

Sydney lies on a beautiful harbour. There are many surfing and non-surfing beaches within easy reach of the city, while the foreshores provide a most pleasant environment. Other attractions are within close proximity to Sydney, including some fine examples of Australia's "bush." Sydney is also an ideal springboard to other Australian destinations, such as the Great Barrier Reef and the Central Australian outback, that may be explored pre and post conference.

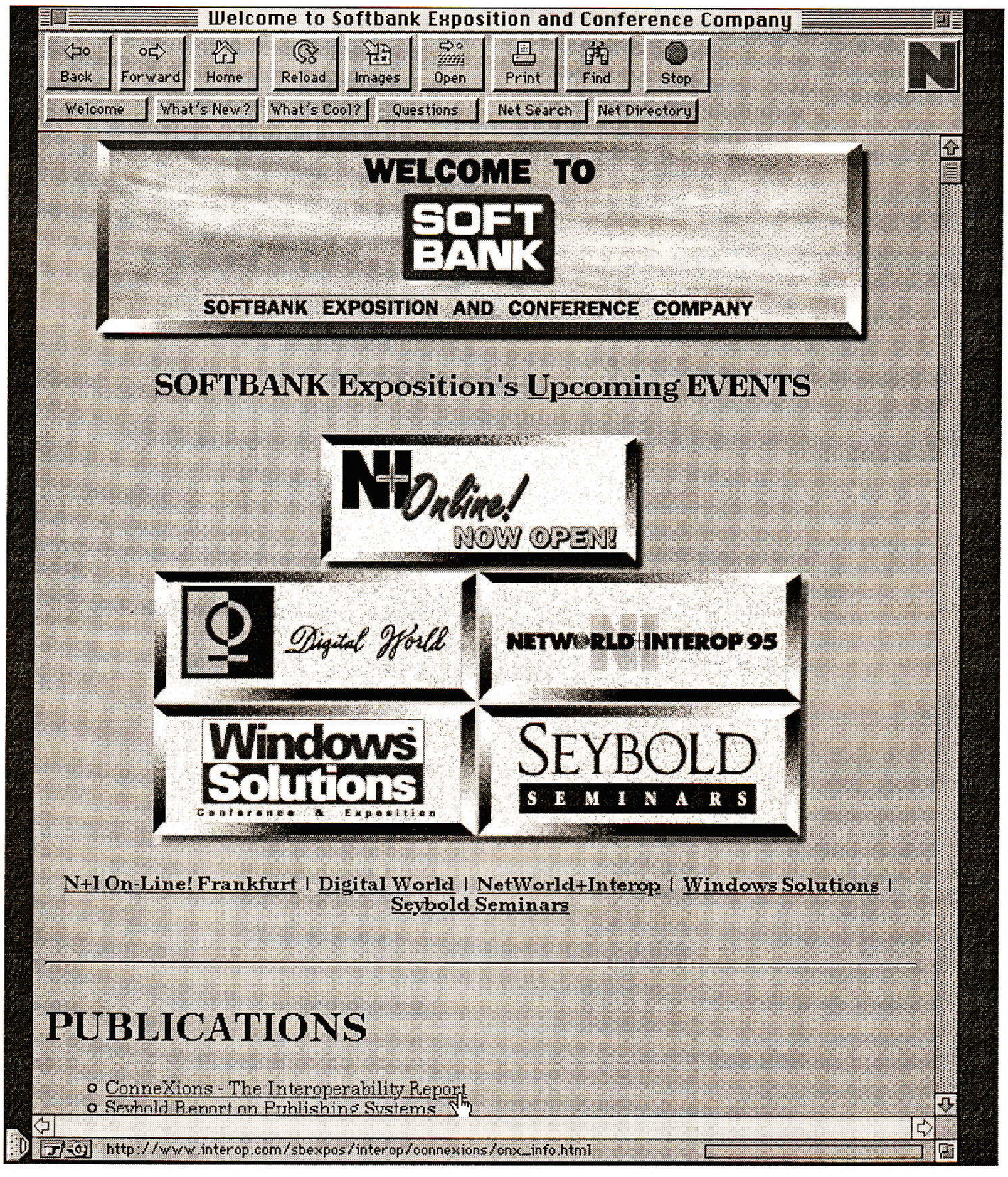
For more information on Australia refer to the URL: `http://www.csu.edu.au/education/australia.html`. December is a great time to visit Sydney. It is early summer, with average temperatures of 23°C.

Related event

Please note that IFIP *Upper Layer Protocols Architectures, and Applications* (ULPAA) Conference will be held directly after the HIPPARCH workshop at the same venue. Information about ULPAA can be obtained from `http://www.ee.uts.edu.au/ifip/ULPAA95.html`.

Come visit our Home Page!

You can get the latest information about NetWorld+Interop as well as all of our other events by connecting to `http://www.interop.com`. You'll also find the most up-to-date *ConneXions* information including a complete index of back issue at the same location. Give it a try!



CONNEXIONS
303 Vintage Park Drive
Suite 201
Foster City, CA 94404-1138
Phone: 415-578-6900
FAX: 415-525-0194

ADDRESS CORRECTION
REQUESTED

FIRST CLASS MAIL
U.S. POSTAGE
PAID
SAN JOSE, CA
PERMIT NO. 1

CONNEXIONS

EDITOR and PUBLISHER Ole J. Jacobsen